Del 1 – Optimering

Lägg CSS i head-taggen
CSS som ligger nära slutet i ett HTML dokument hindrar vissa webbläsare från att börja rendera sidan. Detta på grund av att de väntar på alla css filer ska laddas för att slippa rendera om sidan om någon css ändrar på utseendet. (High performance websites, s. 38)

Lägg Javascript i botten av sidan
När webbläsaren ska ladda in JavaScript så görs detta INTE parallelt. (High performance websites, s. 45-49)

Flytta CSS/JavaScript till externa filer.
Om man har css/js i externa filer kan dessa cachas av webbläsaren. (High performance websites, s. 55-59)

Färre HTTP anrop

- CSS Sprites(High performance websites, s. 13)
- *Kombinera icke ramverk/biblioteks css/js till en fil.*
- *Ta bort döda/oanvända resurs länkar*
- *Minska HTTP-anrop vid hämtning av producenter - Applikationen gör ett anrop för varje producent*
- *Paginering av stora meddelande listor*

Unvika onödiga redirects
Redirects blockerar all data från att laddas ner innan. Detta gör att ingen nedladdning kan påbörjas innan den är klar. (High performance websites, s. 76-79)

Minimera CSS och JavaScript (High performance websites, s. 69-72)

Kombinera anropen till google webfont
Man kan kombinera anrop till google web font iför att slippa extra request (http://googlewebfonts.blogspot.se/2010/09/optimizing-use-of-google-font-api.html)

Om möjligt undvik Etag
Etags skall tas bort om de inte används, annars blir det problem om samma resurs ligger utspridd på olika servrar , de ger då olika värden, vilket mycket väl kan hända i molnlösningar (High performance websites, s. 91-95)

Ajax cache
Vissa AJAX-anrop kan cache:as ur optimeringssyfte. Beror dock på hur man vill ha applikationen.

HTTP cache och gzip
Styrs dock av möjligheten att hantera servern. Kapitel 3 och 4.

Förminska bilders storlek och dimension

Del 2 - Säkerhet

SQL injection
Webbplatsens databas operationer är inte säkra mot SQL injections. Det går alltså att manipulera anrop så att man kan komma åt känsliga uppgifter eller första hela database.

XSS
Ingen av användaren skriven input filtreras och kontolleras.

Session hijacking
Det går att göra en "stjäla" en sessions cookie så att man blir inloggad som den användaren.

Lösenord i databasen är i klartext!
Se föreläsningen

AJAX anrop kräver inte att man är inloggad
functions.php, som sköter ajax-anropen, kontrollerar inte om anropet är auktoriserat, dvs från en inloggad användare.

CSRF
När ett meddelande ska postas så är det möjligt att göra Cross Site Request Forgery.

Användarnamn och lösenord är ifyllt från början/enkla lösenord..

Databas felhantering

Felhanteringen vid databas operationen exponerar databas information som inte borde visas för en slutanvändare.